Bilkent University

Department of Computer Engineering

Senior Design Project

Vortex Sentinel: *Tool for Constructing Website Maps*

Final Report

Supervisor: Uğur Doğrusöz
Jury members: Bedir Tekinerdoğan , David Davenport

Project Group Members:

Serkan Özkul(20601353) Fakih Karademir(20602294)

Mehmet Yayan (20502090) Can Haznedaroğlu (20602445)

İsmail H. Öztürk(20500771)

Contents

1. INTRODUCTION

The tool that we have been trying to design and implement is to construct visual website maps. It is called Vortex Sentinel. Its basic purpose is simple: presenting an elegant web-crawler tool for a wide variety of users. The application is considered to take website URLs as input. After crawling process, internal web documents such as HTML, JavaScript, PHP files and internal references i.e. the links within HTML <a> tags will be represented as the graphical components of a visual website map. Users and webmasters will be able to overview websites, see the documents (nodes) and the links (edges) distribution. The tool should facilitate the works of webmasters, especially. They will be able to view the link statuses between pages, e.g. they will have the chance of viewing broken links. Another important facility expected from our tool is that it should list the e-mail references within a particular website so that the authoritative users can take action for preventing spam mails. Multimedia object list (videos, SWFs, MP3s etc.) within a web document such as HTML file will also be provided via Vortex Sentinel tool. In our application, the graph to be displayed after crawling a particular website will be consisting of nodes which will represent the inter-referenced website objects (e.g. HTML, PHP, JPEG files etc.) and edges which will represent the references among these objects. This graph is the main output of our application, so it should be displayed well and designed consistently. When we think of large websites to be crawled, the graph should definitely have a well designed layout algorithm to display the nodes and edges effectively. For constructing graphs in a proper way, we will use a layout scheme, which is available to our project group by our supervisor. It is called CoSe layout.

The crawling process, generation of the graph layout and dynamic actions performed by users such as adding a new node should be accomplished in a reasonable time interval. These processed are considered to be made as fast as possible. The system should also answer to the dynamic changes in the visualization part at high speed. The application is considered to work in almost all platforms, which have the necessary supporting components such as Adobe Flash displaying and a qualified web browser. The crawling process and generation of information set for constructing graphs are considered to be server-side operations. Actually, this will provide

more efficiency rather than a desktop application. Our system will have good user interfaces to facilitate the user activities.

1.2 Comparisons with the Existing Systems

Although the concept of “Web Crawling” is a well-known and comprehensive topic in computer technology sector, there is no widely used and known software which provides various requirements in this area. There are some software tools like WebSphinx and PHPCrawler which work according to different aspects such as keyword searching. These crawlers have some deficiencies as follows;

• Graphical representations of links are not good enough and comprehensive.

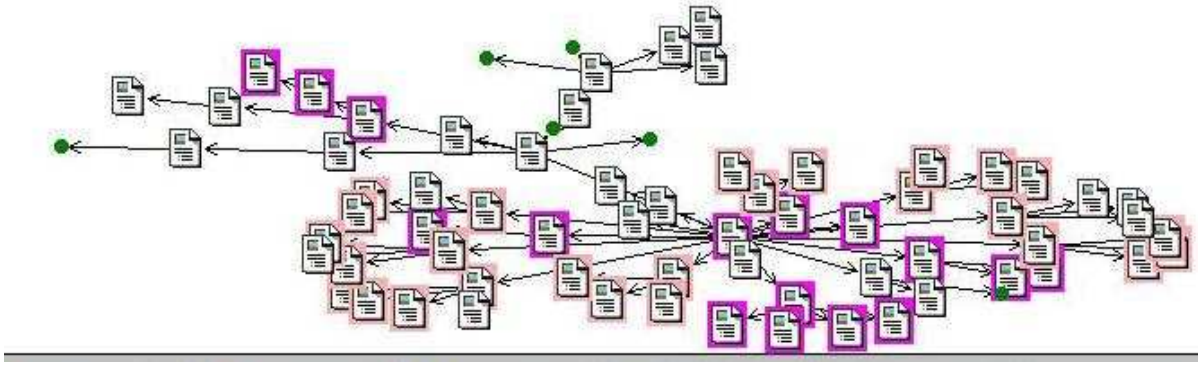


Figure 1: A visual link representation of WebSphinx crawler tool

They do not give the whole link map because of the inefficient Algorithms

.

- Some of them are not recursive so some links and sites cannot be visible.

- They spend too much time (usually in minutes) for crawling and representing visually especially in big websites.

- In most of them there is no search option which provides efficiency and earns time to user.

- They spend too much computer memory (RAM) for crawling and for example after a while user do not work on another thing.

- Generally they do not have user friendly interface, their menus are ineffective and limited.

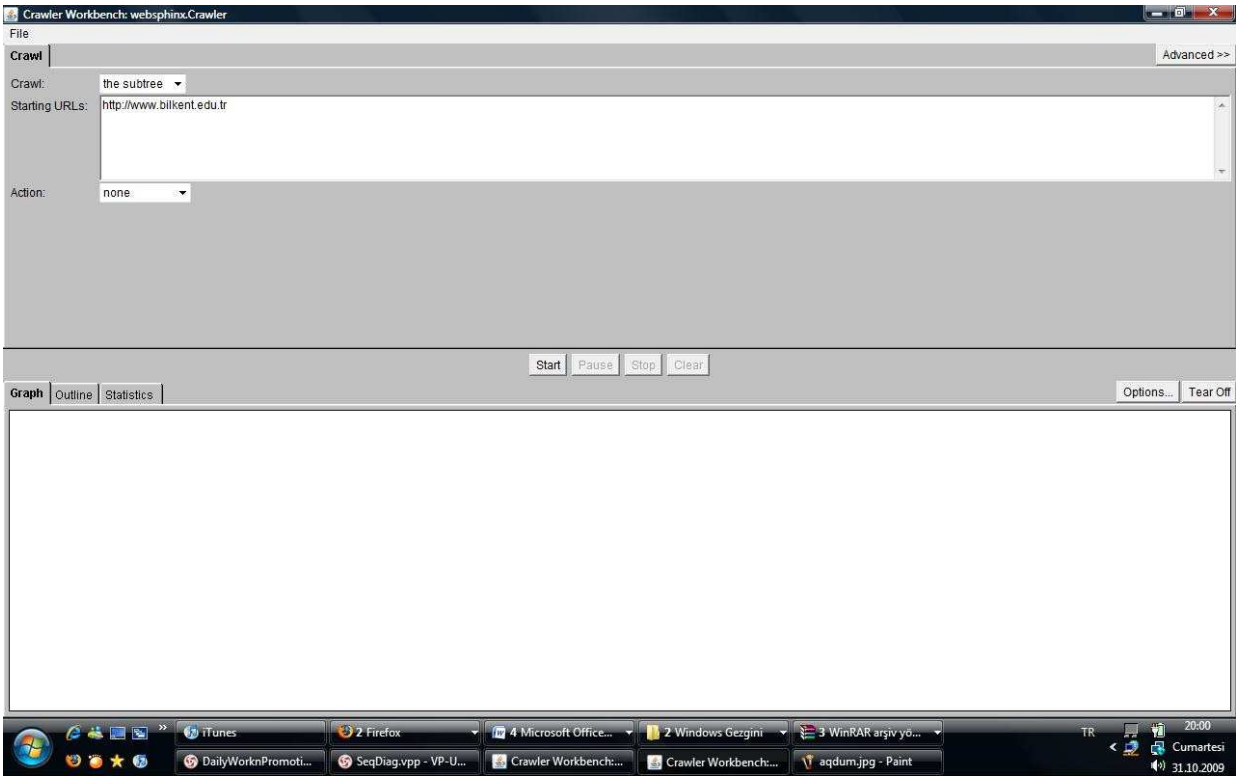


Figure 2: Menu of Web Sphinx crawler tool

- They are platform dependent and most of them work on Windows environment so they cannot be suitable for every user.

- Most of them work as executable file so they require downloading and executing in each time.

- They only show links in web pages and some little features however they do not have “updating” and “fixing” invalid link options. In other words they are likely tools which can do only crawling and show the links but there is a missing point that they should also update links and be able to fix them.

- Because of their limited purposes, they are not able showing various information about web sites like site rankings or thumbnails.

According to our observations from other crawler tools, they have significant deficiencies and also their use cases are so limited. At this point by observing basic deficiencies of these crawlers we determined fundamental features and use cases of our tool and also extra abilities which we added.

Most crawler applications are designed and implemented with the following components. The relations between the components are tentative.

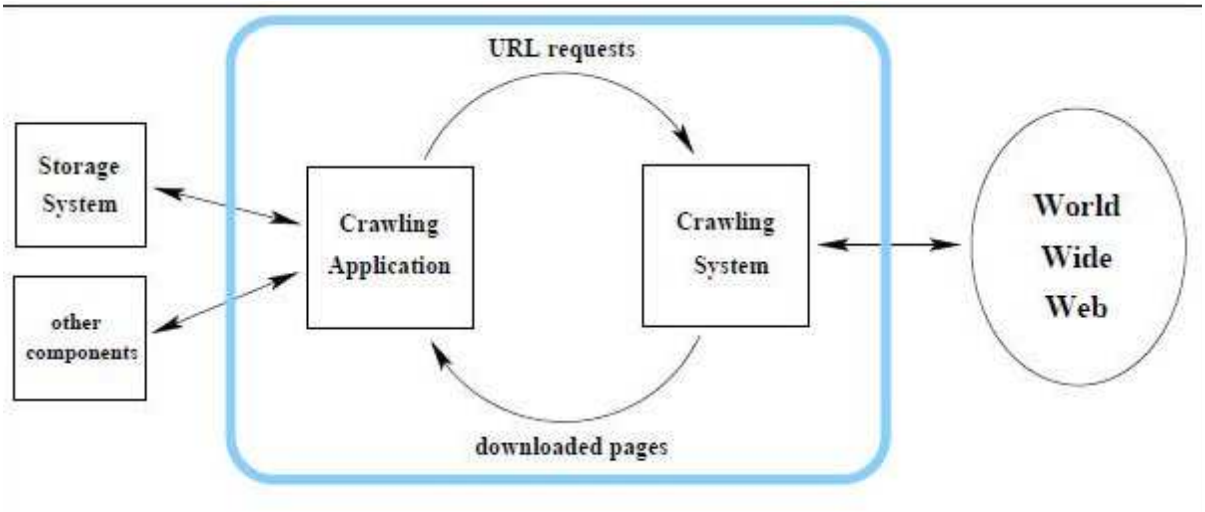


Figure 3: Crawler components

2. ARCHITECTURE AND DESIGN

2.1 General View of Packages

Our project is composed of three major packages that are Database Management Package, Crawler Package and GUI Package. The packages and their relation can be seen in Figure 4.

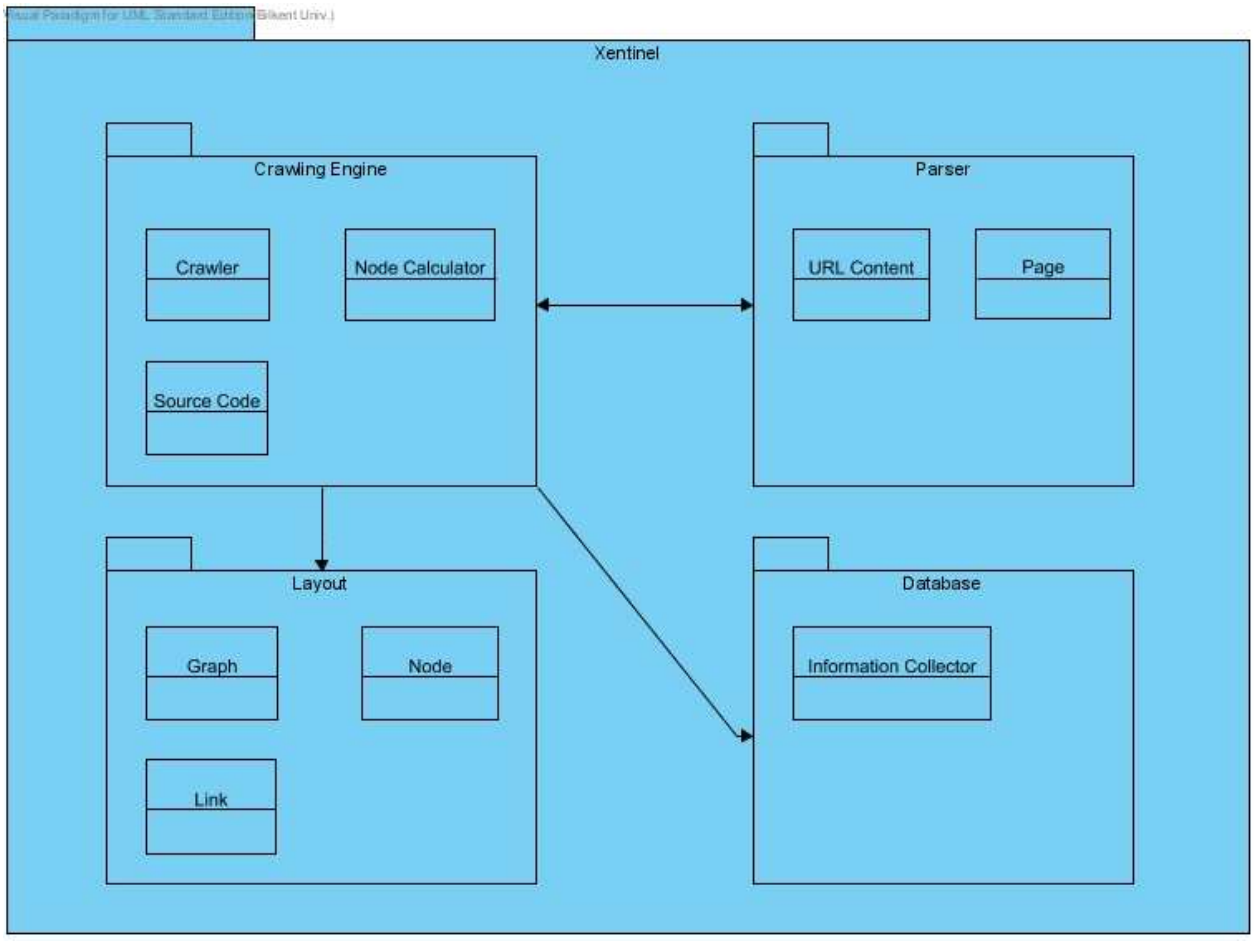


Figure 4: Package Diagram

2.1.1 Database Package

Database Package includes Information Collector Class which is responsible for holding node information in a database. The table structure of the database is shown below;

```
CREATE TABLE `evaturke_vs`.`site_info` (
        `url` VARCHAR( 30 ) NOT NULL ,
        `name` VARCHAR( 30 ) NOT NULL ,
        `links` INT( 10 ) NOT NULL ,
        `xml_file` VARCHAR( 15 ) NOT NULL ,
        `crawl_time` INT( 5 ) NOT NULL ,
        `last_crawled` DATE NOT NULL ,
        `extra_information` VARCHAR( 150 ) NOT NULL
) ENGINE = MYISAM ;
```

Field	Type	Collation	Attributes	Null	Default	Extra	Action
	url	varchar(30)	utf8_unicode_ci		No	None	
	name	varchar(30)	utf8_unicode_ci		No	None	
	links	int(10)			No	None	
	xml_file	varchar(15)	utf8_unicode_ci		No	None	
	crawl_time	int(5)			No	None	
	last_crawled	date			No	None	
	extra_information	varchar(150)	utf8_unicode_ci		No	None	

Figure 5: Structure of site_info table

Attributes:

url: States the URL of the node.

name: Indicates the name of the node.

links: Indicates the number of the links that the source code of given url contains.

xml_file: XML Node structure file path of the given URL.

crawl_time: Time elapsed while crawling.

last_crawled: the last date of crawling the given URL.

extra_information: Miscellaneous information about the URL.

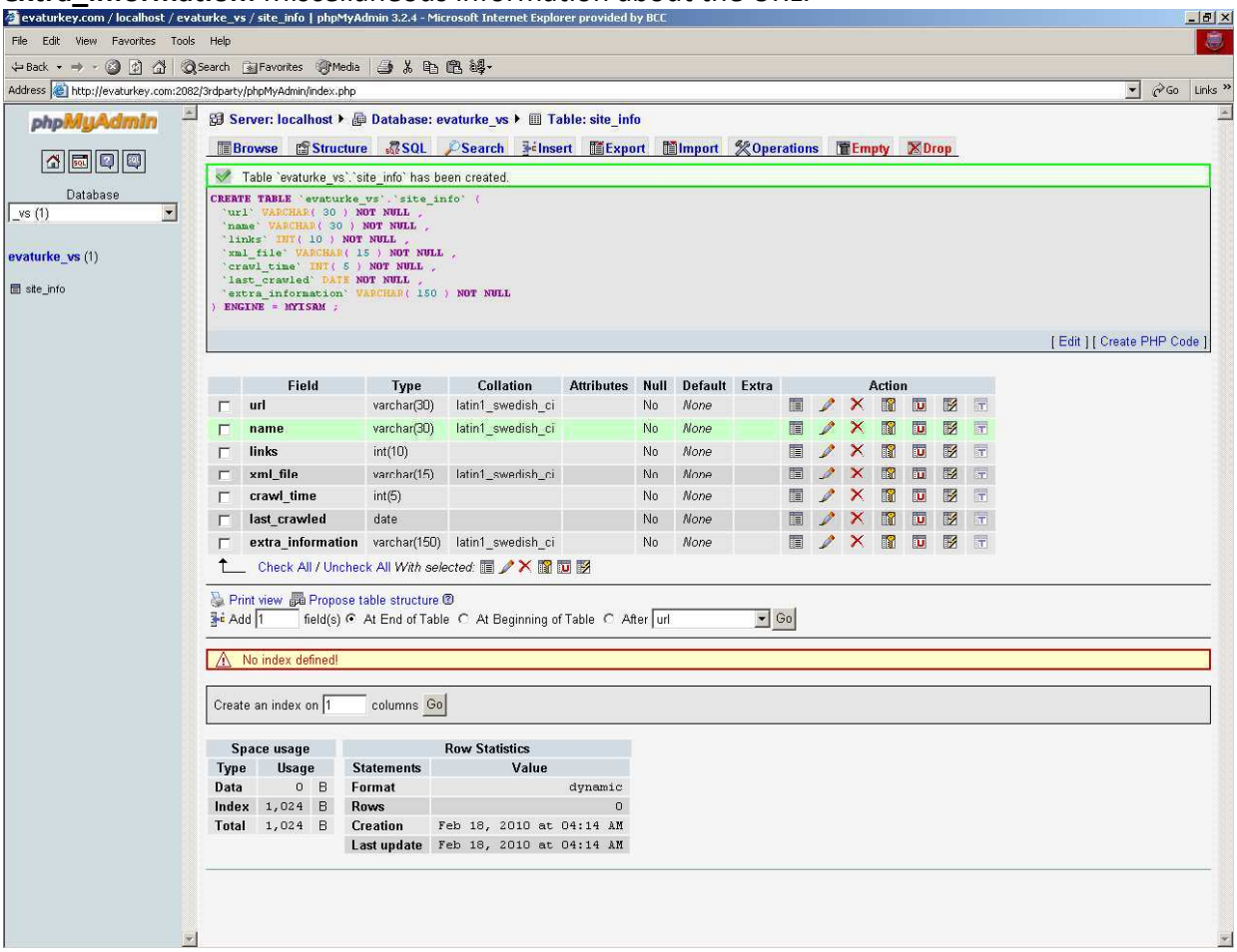


Figure 6: PHPMyAdmin interface about our database

2.1.2 Layout Package

Layout Package includes classes that are combined with flex tool.This package basically manages visual components of the project by managing the creation of nodes and links in the graph. It reads the data sent by PHP which holds the node and edge information, then it construct the nodes and edges on Flash Layout. The Integrration

between PHP and Flash Layout is critical that we have used many resources to accomplish ,that we have. It requires advanced engineering skills that we got successful on this goal but another senior group of Ugur Dogrusöz, who are using the same graphical layout tool, could not make the integration successfully.

2.1.3.Parser Package

Parser Package includes classes that are used to search for links in given URL. This package generally responsible for parsing all possible URLs. Additional feature of that package is parsing the source code the find any search key that is given by the user. Also it can find media files(video, music, etc.), word documents and mail addresses.

2.1.4. Crawling Engine Package

Crawling Engine Package includes classes that are responsible for main crawling by determining recursive crawling levels. It has advanced options that user can set crawling level, crawling algorithm (Breadth First Search or Depth First Search) that increases the functionality. Crawling Engine Package works with the Parser Package that Crawling Engine Package send the html source code to Parser Package to find URL's included in that source code. Then Parser Package send these URL's the Crawling Engine Package and it adds these links the a queue structure to keep crawling process. After that Crawling Engine Package selects the next URL in the queue and fetches its source code then sends it to Parser Package to do the same process recursively.

You can examine the component diagram of Xentinel tool for better understanding:

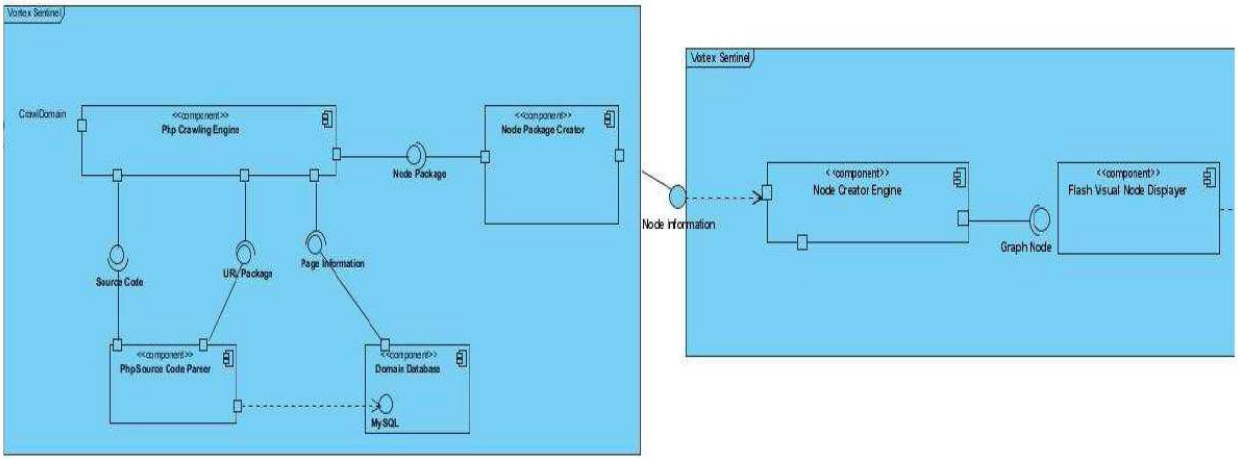
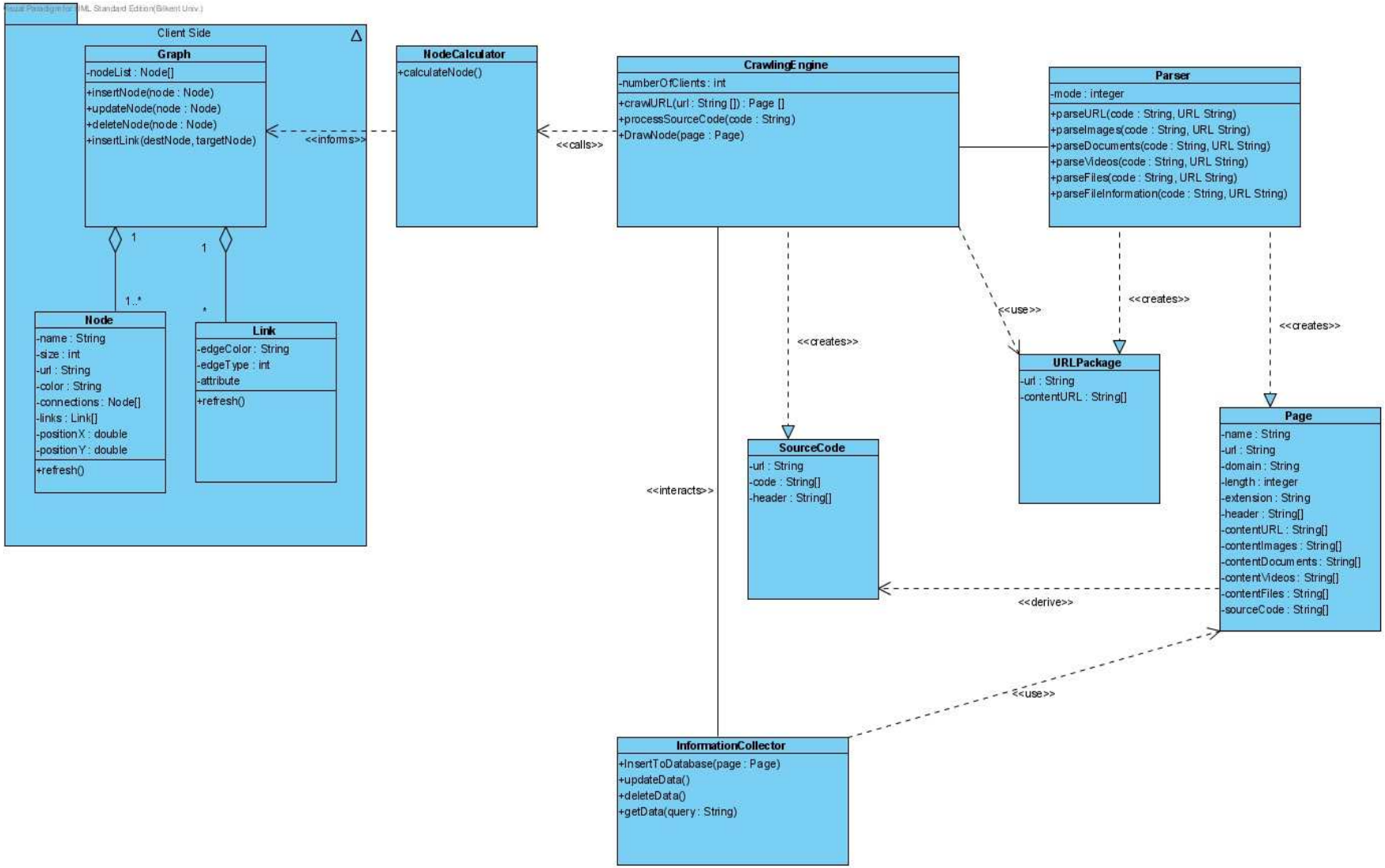


Figure 5: Component Diagram

When Vortex Sentinel applicaiton is launched, the users are expected to enter input URL. CrawlingEngine takes this URL and tries to extract the source code of the base file (e.g. index.html) to trigger the actual crawling process. Then it sends the generated code to URLParser. Regular updates are made in related parts of database components. CrawlingEngine also collects the cumulative data to determine and form node packages which include parameters like node info, edge info etc. These packages are sent to LayoutManager and via VisualMapManager overall website map is displayed.

The general class diagram of our proposed solution tool is as follows:


Client Side
Graph
-nodeList : Node[]
+insertNode(node : Node)
+updateNode(node : Node)
+deleteNode(node : Node)
+insertLink(destNode, targetNode)
<<informs>>
NodeCalculator
+calculateNode()
<<calls>>
CrawlingEngine
-numberOfClients : int
+crawlURL(url : String []) : Page []
+processSourceCode(code : String)
+DrawNode(page : Page)
Parser
-mode : integer
+parseURL(code : String, URL String)
+parseImages(code : String, URL String)
+parseDocuments(code : String, URL String)
+parseVideos(code : String, URL String)
+parseFiles(code : String, URL String)
+parseFileInformation(code : String, URL String)
1
1
1..*
*
Node
-name : String
-size : int
-url : String
-color : String
-connections : Node[]
-links : Link[]
-positionX : double
-positionY : double
+refresh()
Link
-edgeColor : String
-edgeType : int
-attribute
+refresh()
<<use>>
<<creates>>
<<creates>>
<<creates>>
URLPackage
-url : String
-contentURL : String[]
SourceCode
-url : String
-code : String[]
-header : String[]
Page
-name : String
-url : String
-domain : String
-length : integer
-extension : String
-header : String[]
-contentURL : String[]
-contentImages : String[]
-contentDocuments : String[]
-contentVideos : String[]
-contentFiles : String[]
-sourceCode : String[]
<<interacts>>
<<derive>>
<<use>>
InformationCollector
+InsertToDatabase(page : Page)
+updateData()
+deleteData()
+getData(query : String)

2.2 Class Documentation

2.2.1 Interface Documentation Guidelines

The format of the class documentation is located below;

Class	Name of the Class
Type	Type of the class
Description	Brief description of the class that includes its functionality
Package	Package that contains the class
Attributes	Name of the attributes
Operations	Name of hte operation(Name of the parameters): return type

2.2.2 Classes of the System

CrawlingEngine.php

Class	CrawlingEngine.php
Type	PHP Class
Description	
Package	Crawling Engine Package
Attributes	numberOfClients: int

Operations	**crawlUrl (url : String[]) : page[]** **// Crawls the given url and returns the page array** **processSourceCode (code : String) : void** **//Send source code to Parser In order to pars URL's in the source code** **drawNode(page : Page): void** **//draw the node into the page according to given URL's**

Parser.php

Class	Parser.php
Type	PHP Class
Description	**Makes data mining on a given source code to parse specific information.**
Package	Parser Package
Attributes	**mode : integer**
Operations	**parseURL(code : String, URL : String) : String []** **//Parses all URLs (*.html, *.htm, *.php, *.asp) that are included in the given source code.**
	parseImages(code : String, URL : String) : String [] **//Parses all image files (*.jpg, *.jpeg, *.bmp, *.png, *.gif etc) that are included in the given source code.**
	parseDocuments(code : String, URL : String) : String [] **//Parses all document files (*.doc, *.docx, *.ppt, *.pdf) that are included in the source code.**

	parseVideos(code : String, URL : String) : String [] **//Parses all video files (*.avi, etc) that are included in the source code.**
	parseFiles(code : String, URL : String) : String [] **//Parses all other unkown typed files that are included in the source code.**
	parseFileInformation(code : String, URL : String) : String [] **//Parses file information (title, extension, size, meta data, header information etc.) that are included in the source code.**

InformationCollector.php

Class	**InformationCollector.php**
Type	PHP Class
Description	Sends the information of given Page object to database which interacts with Crawling Engine
Package	Database Package
Attributes	None
Operations	insertToDatabase(Page page) //inserts the page object that is given as parameter to database.
	updateData(Page page, String URL) //updates the given page by modifying its URL to the given parameter.
	deleteData(Page Page) //deletes the specified page from database

	getData(String query) //returns data for the specified input query

NodeCalculator.php

Class	NodeCalculator.php
Type	**PHP Class**
Description	**It is responsible for connection between server and client side; also PHP and ActionScript.**
Package	**CrawlingEngine Package**
Attributes	**None**
Operations	**CalculateNode ()** **//Provides integration between PHP and ActionScript.**

URLContent.php

Class	**URLContent.php**
Type	**PHP Class**
Description	**Keeps the contents of the URL package**
Package	**Parser Package**
Attributes	**url :String** **contentURL :Sting[]**
Operations	

Page.php

Class	Page.php
Type	PHP Class
Description	**Page class hold the detailed page information**
Package	Parser Package
Attributes	**name: String** **url: String** **domain: String** **length: int** **extension: String** **header: String[]** **contentURL: String[]** **contentImages: String[]** **contentDocuments: String[]** **contentVideos: String[]** **contentFiles: String[]** **sourceCode: String[]**
Operations	

SourceCode.php

Class	SourceCode.php
Type	PHP Class
Description	Source Code Class stores the unparsed information
Package	Crawling Engine Package
Attributes	**url: String** **code: String[]** **header: String[]**
Operations	

Graph.as

Class	Graph.as
Type	Action Script Class
Description	**It is responsible for creating graphic objects at client side.**
Package	**Layout Package**
Attributes	Nodelist : Node []
Operations	**insertNode (Node node)** **updateNode (Node node)** **//Updates a node**

	deleteNode (Node node) //Deletes a node insertLink (Node destnode , Node targnode)

Node.as

Class	Node.as
Type	Action Script Class
Description	It is responsible for creating nodes at client side.
Package	Layout Package
Attributes	name : String size : int url : String color : String connections : Node [] links : Link [] positionX : double positionY : double
Operations	refresh () //Refreshes a node

Link.as

Class	Link.as
Type	**Action Script Class**
Description	**It is responsible for creating links at client side.**
Package	**Layout Package**
Attributes	**edgeColor : String** **edgeType : int** **attribute**
Operations	**refresh ()** **//Refreshes a link**

3. FINAL STATUS OF THE PROJECT

During two whole semester, our group have completed nearly all requirements that we had proposed with little exceptions. Vortex Sentinel works as inteded by searching links of the given website and transposes them into a connected graph with the help of Flex. Vortex Sentinel is also able to multi level crawl first-hand founded links recursively.

4. IMPACT OF THE ENGINEERING SOLUTION

4.1. Economic Constraints

Our system is able to run with low cost and high performance that we need only a 24 hour online server to serve clients properly. After installation of necessary software to our server, there will not be additional cost to make Vortex Sentinel Online. There may be slight monthly maintenance cost of the server which can be ignored so we can keep performance/cost ratio.

4.2.Environmental Constraints

It has no impact on physical environment but has a considerable effect on digital environment because our system crawls web pages of a domain swiftly so target server which holds that domains should be powerful not to crash.

Vortex Sentinel uses encryption to protect login information of the users and does not store this information after the crawling process. We take care of implementation to prevent any vulnerability in our system.

4.3. Social Constraints

Vortex Sentinel helps user to spend less time on web sites by providing whole site as a connected graph in order to observe easily any particular site.Since the only interaction is between user and computer systems, it has no other social aspect.

4.4. Political Constraints

Our system struggles to make objective crawling as much as possible. Since our system is a useful tool for crawling webistes in order to create a connected graph of the links betweeen them, it has no political constraint.

4.5. Ethical Constraints

Vortex sentinel is responsible of ethical issues while crawling a website thus it can identify pages as not to be crawled by disallowing in a separate text file in order to sustain its responsibility. Vortex sentinel also crawls with the functionality as a design principle and uses several design approaches (divide & conquer approach ,top down strategy) as a design solution. Vortex Sentinel takes account of privacy and ethical rules while crawling process the websites which is made by checking robots.txt first before starting to crawl. Webmasters may put a “robots.txt” file in to the root folder of their websites to indicate which url’s desired not to be crawled.

A sample robots.txt file;

www.ornek.com için robots.txt dosyası
User-agent: *
Disallow: /cgi-bin/
Disallow: /images/private/
Disallow: /private.html

As we see from the example above, robots.txt file states that “/cgi-bin/” folder, “/images/private/” folder and “/private.html” file should not be crawled for privacy and security reasons. So Vortex Sentinel considers robots.txt file to follow ethical rules.

4.6. Health Constraints

Our system does not constitute any impairment to health. Feel free to use but no more than 10 hours.

4.7. Safety

Vortex Sentinel uses encryption to protect login information of the users and does not store this information after the crawling process. We take care of implementation to prevent any vulnerability in our system.

4.8. Manufacturability

Since it is software project which is desired to be installed into a server and expected to be run on a server, it has no manufacturability aspect since we decided not to make an additional client version of Vortex Sentinel that runs on client pc and does not require a server, it has pretty high manufacturability aspect and does not require any cost.

4.9. Sustainability

Sustainability of Vortex Sentinel is quite satisfying that keeps serving to its clients without crashing. We may limit the maximum of users served at a time according to results of stress testing so our system would not be crashed because of denial of service on the other hand we use a database system to keep track of website information which requires to be optimized monthly to stabilize performance.

4.10. Professional and Ethical Responsibility

Vortex sentinel is responsible of ethical issues while crawling a website thusit can identify pages as not to be crawled by disallowing in a separate text file in order to sustain its responsibility. Vortex sentinel also crawls with the functionality as a design principle and uses several design approaches (divide & conquer approach ,top down strategy) as a design solution.

4.11. Low Cost and High Performance

Our system is improved to be used for crawling up to several hundred pages per second which leads to millions of pages per run. System is also run on low-cost hardware. It is extremely crucial that efficient use of disk access provides high speed with the help of main data structures such as the structure that involves URL seen. This situation can only occur when crawling several million pages.

4.12. Robustness

Since the system interacts with several millions of servers, it is developed to be reliable against bad HTML and broken links, strange server behavior and configurations and many other situations that involve crawling errors. Thus, the goal here is to avoid as many broken links and bad requests as possible since in many applications program is going to download a subset of pages anyway. Also the system is desired to be tolerant against any computer crashes or network interruptions since a crawler can take days or weeks. Thus, in any time the state of the system is kept on disk. Since system does not require strict ACID properties, it is appropriate that periodic synchronization of the main structure to disk should be used.

4.13. Etiquette and Speed Control

System was designed to be able to control access speed in several different ways. We have to avoid putting too much load on a single server; we do this by contacting each site only once second unless specified otherwise. It is also desirable to throttle the speed on a domain level, in order not to overload small domains, and for other reasons to be explained later. Finally, since we are in a campus environment where our connection is shared with many other users, we also need to control the total download rate of our crawler. Also, crawling at low speed during the peak usage hours of the day, and at a much higher speed during the late night and early morning, limited mainly by the load tolerated by our main campus router. To control the speed, we added a crawling speed controller which sleeps the crawler after fetching the

html source code of every page, another fact that we have limited the number of users (5) can get service from Xentinal at the same time.

4.14. Manageability and Reconfigurability

Suitable interface for monitoring the crawler is provided by the hosting company, including the speed of the crawler and the sizes of the main data sets with the statistics about hosts and pages. Admin is able to alter the speed and have the option of adding and removing components, shutting down the system, forcing a checkpoint and adding hosts that include broken links or bad requests to the list of places that the crawler should avoid. System is modified after any crush or shutdown and fixs any problems that occur in order to continue crawling by using different machine configuration.

4.15. Novel Solutions to Accomplish Project

We have used divide-and-conquer approach to provide a design solution. We have carried out design principles such as unity, harmony and functionality during our project analysis and design. We have used URL, e-mail and multimedia object identification and normalization while constructing a web crawler architecture. In the project reports, we have provided mock-ups telling about the design principles used for identifying visual components such as color, line type, texture.

We have used a novel solution to the design problem of e-mail and multimedia file extraction from source code of web-pages which is not a functionality of typical web crawlers. Also there were a design problem related with recording website information, performance, saving backups of websites for future usage and implementing extra features, we used a novel solution by adding a database management system to accomplish these problems. So we can keep track of every website, save previously crawled website information, increase performance of Vortex Sentinel and implement additional features easily.

5. CONTEMPERORY ISSUES ABOUT AREA OF THE PROJECT

Xentinel (Vortex Sentinel) is a web-based crawling engine to get information about websites and construct visual web maps. It crawls the given website, stores the information about website in database and construct a visual web map of that site. Besides Xentinel gives support of making search queries on that website. To compete with other crawlers, our crawling engine should have features of fast crawling, reliable resulting every time, fast and proper parsing of the source code, also should be robust while crawling because scanning a website is a complex and long process that website can have thousands of pages and each page can have many lines of html source code that, we have search all of them and mine that source codes. Todays most popular searching engine Google is the outstanding at many of these issues, it makes very fast searching among billions of web pages and bring result to you in a few milliseconds. Our crawling engine gives a satisfying result on most of these issues, and is improved on performance to give a quick and proper result. As an extra feature, our crawling engine has fast searching feature that user can make querires on our database and get a quick result. On that matter, a new featrre, semantic searching, is become popular among all search engines. Semantic searching makes a search not only based only the query word, it also makes another queries about the meaning of that word so that semantic search gives you more accurate results. At that moment, there is no crawling engine which integrated semantic search completely. To compete with other crawlers, we designed semantic search feature that our server keeps a dictionary to get semantic of given word to search and makes multiple searches besides from the given word, it also makes searches about 3 most accurate meaning of that word, them merges them all. So Xentinel can give more accurate search results to user. Another point is the safety and security that we implemented Xentinel in the safest way that database access is protected and tracked carefully so that system promise reliability and robustness.

6. TOOLS AND TECHNOLOGIES USED

6.1. Adobe Flex Builder 3

Adobe Flex is a software development kit released by Adobe Systems for the development and deployment of cross-platform rich Internet applications based on the Adobe Flash platform. Flex applications can be written using Adobe Flash Builder or by using the freely available Flex compiler from Adobe.

We have used Flex Builder 3 for visualizing the web pages. After filtering the html source of a page, we construct the nodes from the links found, then we sent these data to the Flex Builder and see the nodes and edges, basically the relations, interactions can be done with nodes by changing size, position, color, label or you can connect or disconnect any node and edges that you want.

6.2. Swish Max 3

SWiSH Max is a flash creation tool that is commonly used to create interactive and cross-platform movies, animations, and presentations. It is developed and distributed by Swishzone.com Pty Ltd, based in Sydney, Australia. SWiSH Max primarily outputs to the .swf format, which is currently under control of Adobe Systems that we used the improve graphical layout and effects of our system.

6.3. Rapid PHP Editor

Rapid PHP editor is a powerful, quick and sophisticated PHP editor with features of a fully-loaded PHP IDE and speed of the Notepad. Convenient features enable you to instantly create and edit not only PHP, but also HTML, XHTML, CSS and JavaScript code, while integrated tools allow you to easily debug, validate, reuse, navigate and format source code. We have used Rapid PHP Editor to implement server side of Xentinel which includes integration between PHP and Flash layout.

6.4. WireShark Network Sniffer

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark is cross-platform, using the GTK+ widget toolkit to implement its user interface, and using pcap to capture packets; it runs on various Unix-like operating systems including Linux, Mac OS X, BSD, and Solaris, and on Microsoft Windows. We have used WireShark to sniff data transfer between Flash layout and the Chisio Web Server and tested whether data is transmitted successfully.

6.5. Apache Server with Cpanel 11 Interface

We have used Apache server with cpanel interface to deploy our server-side files and to offer service to users. Also we have monitored and tested our system by using the features of cpanel.

cPanel is a Unix based web hosting control panel that provides a graphical interface and automation tools designed to simplify the process of hosting a web site. cPanel utilizes a 3 tier structure that provides functionality for administrators, resellers, and end-user website owners to control the various aspects of website and server administration through a standard web browser.

6.6. PHPMyAdmin

phpMyAdmin is an open source tool written in PHP intended to handle the administration of MySQL over the World Wide Web. It can perform various tasks such as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions. We have used PHPMyAdmin to make database integration of Xentinel and to test if our system works correctly.

6.7. Webalizer

The Webalizer is a GPL application that generates web pages of analysis, from access and usage logs, i.e. it is web log analysis software. It is one of the most commonly used web server administration tools. It was initiated by Bradford L. Barrett in 1997. Statistics commonly reported by Webalizer include: hits; visits; referrers; the visitors' countries; and the amount of data downloaded. These statistics can be viewed graphically and presented by different time frames, such as per day, hour, or month. We have used Webalizer in the testing phase of our system which involves crawling speed control and to improve our crawling engine.

6.8. Ulead PhotoExpress

Ulead PhotoExpress is a graphics editing tool which is developed by Ulead Company. We have used that tool to design backgrounds, button icons and graphical effects

7. USE OF RESOURCES

During design and implementation of our project, we found beneficial information on some resources. There are 2 main kinds of resources; these are open source resources such as websites, and the books which are related about ActionScript 3.0 and PHP. Beside our experienced friends about ActionScript and PHP, helped us when we had a problem about implementation.

7.1. Open Sources

During our project implementation, we mainly use Internet resources to get any kind of help and idea. We mainly used the official site of PHP and ActionScript. In addition to these, in PHP forums there are many people, who are really interested in PHP applications and have some problems about PHP, and we utilized them during implementation.

7.2. Books

Moreover, we used some books which are related about ActionScript 3.0 and PHP during implementation of our project. Some books are;

- PHP Bible, 2nd Edition by Tim Converse and Joyce Park
- Programming PHP by Rasmus Lerdorf, Kevin Tatroe, and Peter MacIntyre
- ActionScript 3.0 Cookbook: Solutions for Flash Platform and Flex Application Developers by Joey Lott, Darron Schall, and Keith Peters
- ActionScript 3.0 Bible by Roger Braunstein

7.3. Library Resources and Internet resources used

We have used Adobe Flash CS4 which our instructor suggested us in order to better implement graph component of our web crawler by using its graph library and its filter library.

We also used HTTP and filtering libraries of PHP in order to extract the URL from the source code and mySQL libraries in order to communicate with the database which we keep all the website information. So that we have used mainly developer forums for ActionScript and PHP to aid us in our project. You can find these forums the references section.

8. GLOSSARY

Term	Explanation
Vortex Sentinel	The name of our system and tool.
User	The person who uses the Vortex Sentinel system via a Web browser
Site administrator	The manager of a website who wants to see the map of the site via our tool
Node	Graphical element that represents a linked or referenced Web object within a particular website (e.g. html file, jsp file or jpeg image)
Edge	Graphical element that represents the connection between two Web documents within a particular website
XML	(Extensible Markup Language) designed to transport and store data.
GraphML	(Graphical Markup Language) used to describe the structural properties of a graph and a flexible extension mechanism to add application-specific data.
AS 3.0	ActionScript 3.0 is a flash scripting language.
PHP	HyperText Preprocessor: general purpose scripting language.
Parser	Parser is a component in our project which scans and mines the html source code to filter undesired content.

9. CONCLUSION

Vortex Sentinel is a web crawler tool that aims to present a website as a graphical map. We have tried to design and implement the tool within the scope of this purpose. The system works with a graphical framework support and crawler part. The graphical support tool was provided to us so that we can use some layouts for the website maps. And we have implemented web crawler part with the help of PHP programming language. The communication between the crawler and graphical parts was yielded with the help of XML technology.

For the crawler part, we have used a crawler algorithm which starts from a webpage then finds the links coming from this page. Then the founded pages processed with the same operation. And visited pages are kept in order to prevent redundant crawling operations.

The graphical part takes the website map elements from an external XML file which is produced by the crawler part at runtime. The specification in the XML file read and accordingly the graph or the map of the website is presented. Nodes and edges are placed with relative information such as the name of the pages.

In this final report, we have mentioned about the final process throughout the project. Improvements in the tool are included. System architectures are given. Software packages are given and explained.

10. REFERENCES

1] IVIS, Bilkent University CS Department
http://www.cs.bilkent.edu.tr/~ivis/layout-demo/lw1x.html

[2] Web Crawler, Wikipedia
http://en.wikipedia.org/wiki/Web_crawler

[3] WebSPHINX, Carnegie Mellon University CS Department
http://www.cs.cmu.edu/~rcm/websphinx/

[4] Web Crawler, Polytechnic Institute of New York University CS Department
http://cis.poly.edu/tr/tr-cis-2001-03.pdf

[5] Crawling the Web, University of IOWA
http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf

[6] Deep-Web Crawl, Cornell University CS Department
http://www.cs.cornell.edu/~lucja/Publications/i03.pdf

[7] Deep Web, Wikipedia
http://en.wikipedia.org/wiki/Deep_Web

[8] Focused Crawling, Indian Institute of Technology Bombay Department of CS&E
http://www.cse.iitb.ac.in/~soumen/focus/

[9] EffectiveWeb Crawling, University of Chile CS Department
http://www.chato.cl/papers/crawling_thesis/effective_web_crawling.pdf

[10] Distributed Web Crawling, Wikipedia
http://en.wikipedia.org/wiki/Distributed_web_crawling

[11] Extensible Web Crawler, University of Illinois MIAS
http://www.mias.uiuc.edu/files/tutorials/mercator.pdf

[12] DevNetwork Forums
http://forums.devnetwork.net/

[13] Dev Shed Forums
http://forums.devshed.com/

[14] Adobe Flex
http://en.wikipedia.org/wiki/Adobe_Flex

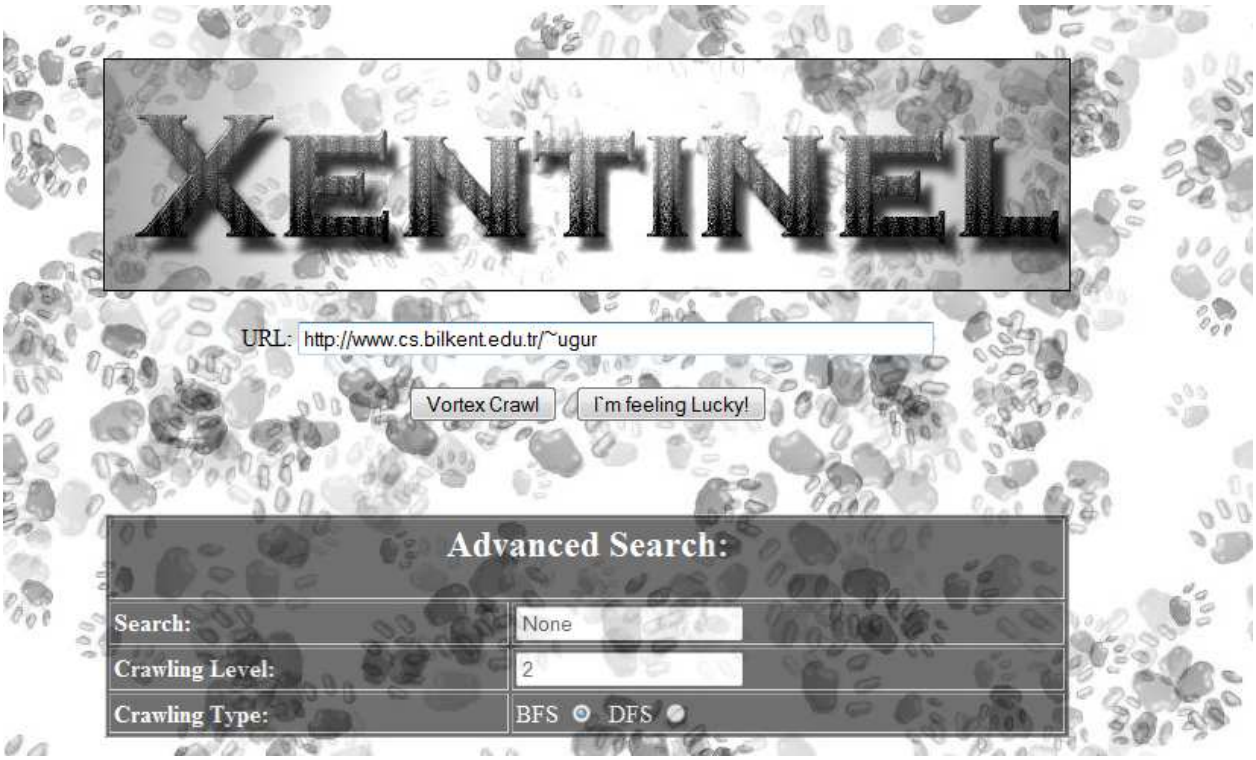
11. APPENDIX

11.1. User Manual

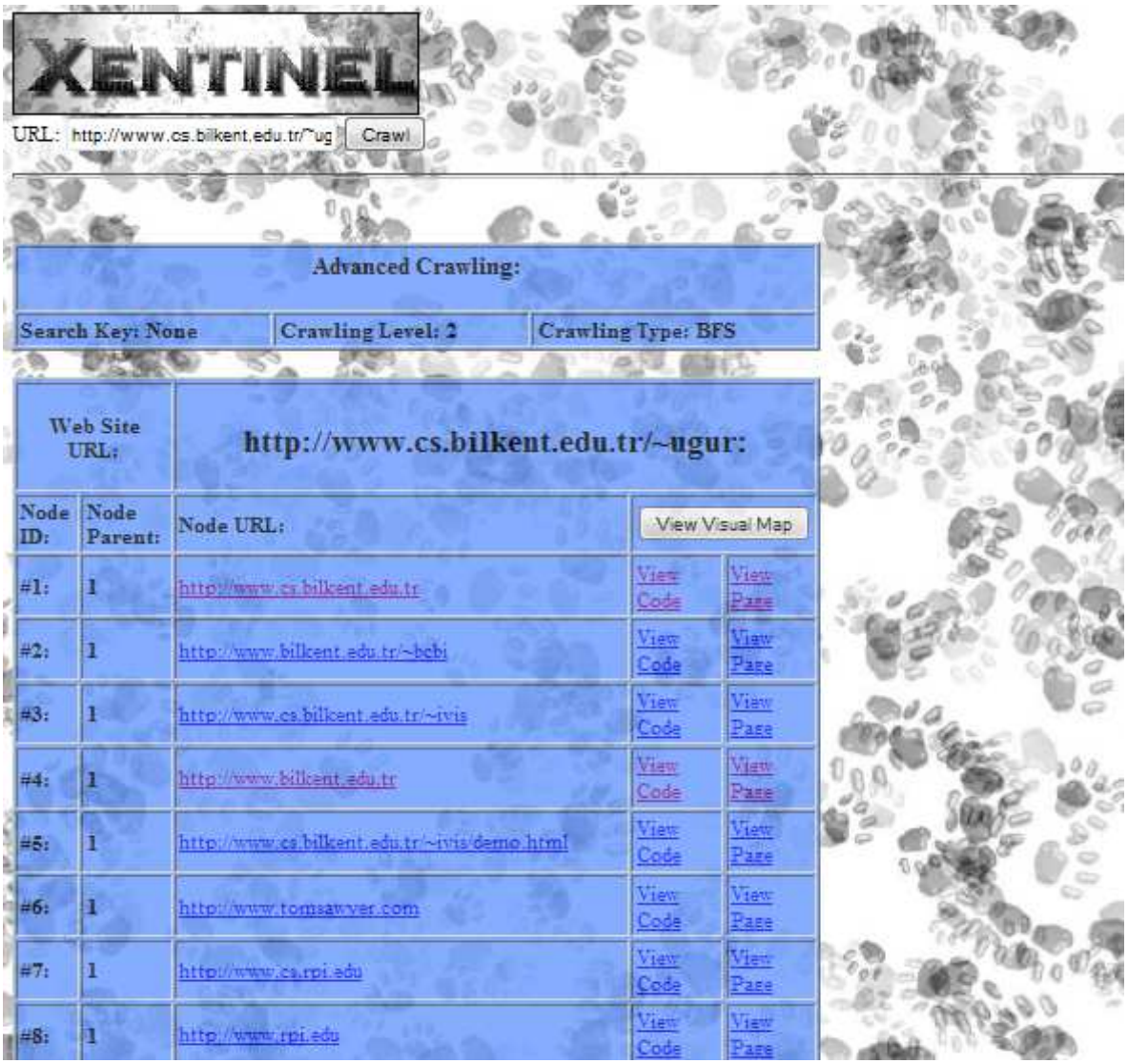
When user opened the website of our crawler,

http://www.evaturkey.com/CS491/crawler/

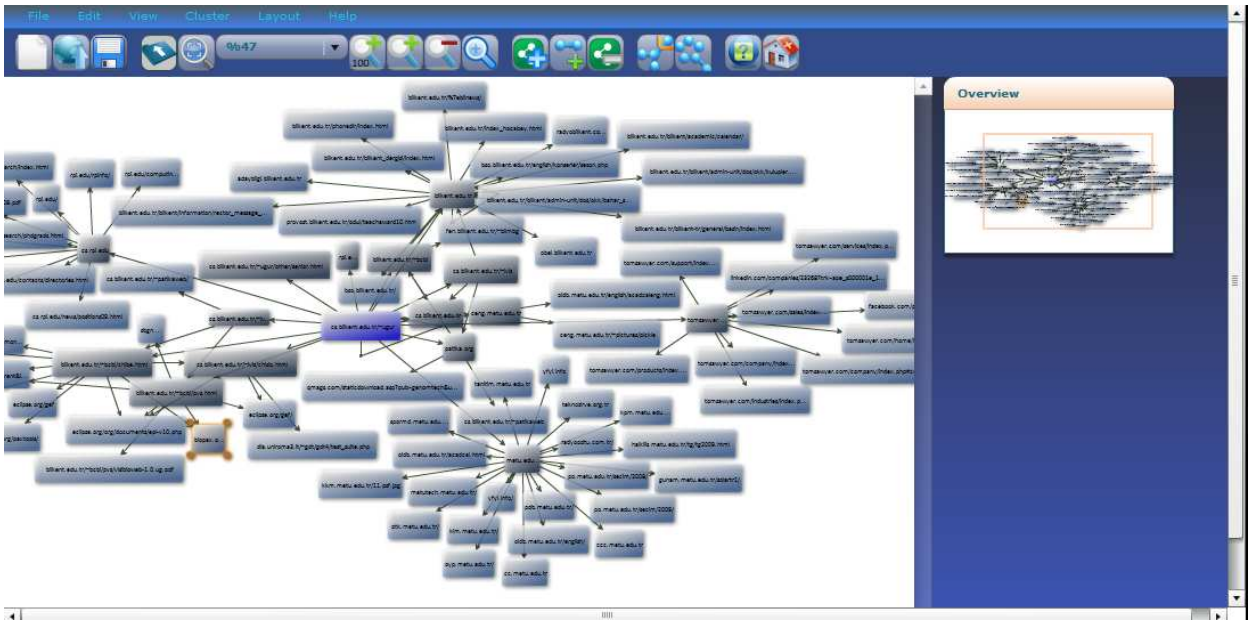
He can type down the URL he wants to crawl and chooses any advanced search option he wants. User may search for a keyword in the crawled pages, he may adjust the recursive crawling level and he may change the crawling type to either Breadth-First-Search or Depth-First-Search as advanced crawling options.



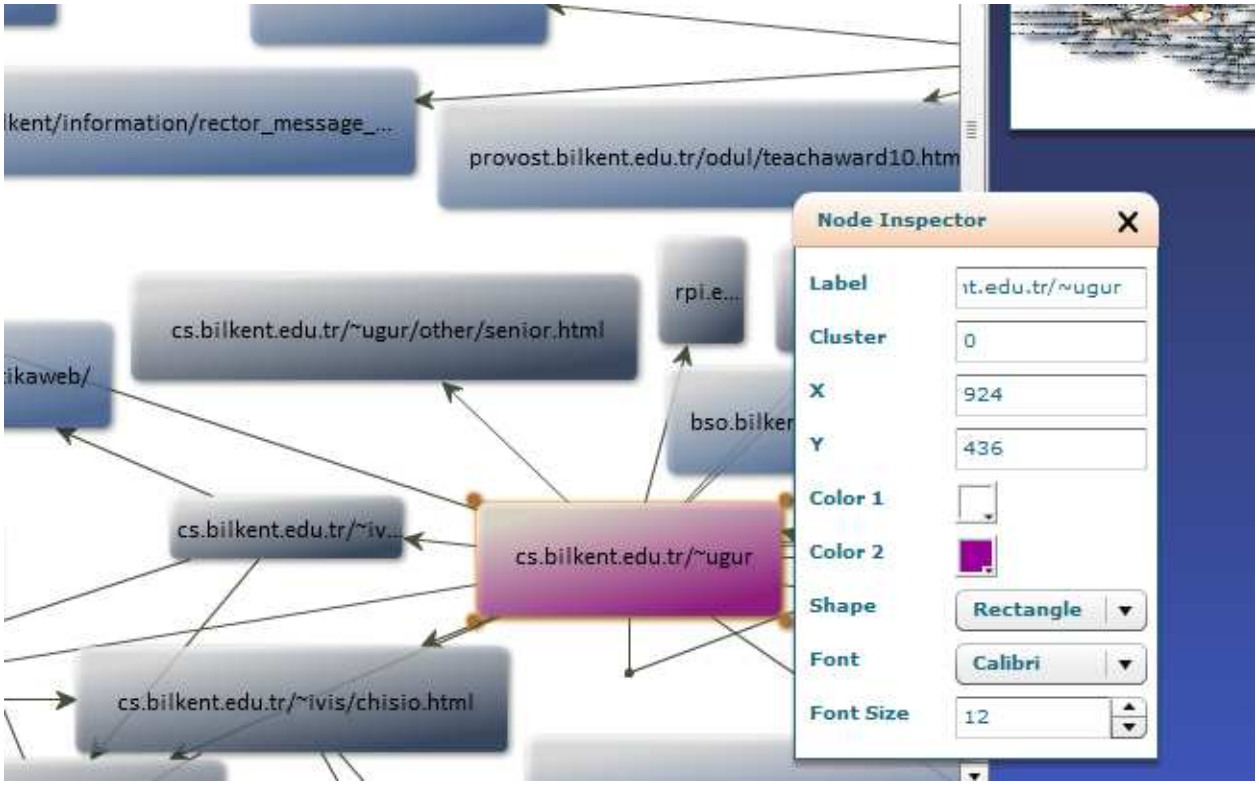
After user chooses his advanced crawling options and clicks Vortex Crawl button, our crawler goes to the URL that is given as input and starts find links to other URLs and lists all URLs that it has found to our website.



Most importantly, our crawler can visualize found URLs as connected graph rather than lists if user clicks View Visual Map. Once user views the map new window appears which contains the connected graph of the crawled URLs.



User may also inspect any node with node inspector by double clicking on that particular node.



A mock-up screenshot that we designed at the beginning of our project that you can compare we designed and implemented.